

Responsive Graphical User Interface (ReGUI) and Its Implementation in MATLAB

Matej Mikulsky
Jana Pocsova
Andrea Mojzisova
Igor Podlubny*

BERG Faculty, Technical University of Kosice,
B. Nemcovej 3, 04200 Kosice, Slovakia

matej.mikulsky@student.tuke.sk,
jana.pocsova@tuke.sk
andrea.mojzisova@tuke.sk
igor.podlubny@tuke.sk

April 13, 2017

Abstract

In this paper we introduce the responsive graphical user interface (ReGUI) approach to creating applications, and demonstrate how this approach can be implemented in MATLAB. The same general technique can be used in other programming languages.

1 Introduction

The term *Responsive Web Design*, or RWD, was introduced by Ethan Marcotte in his online article in 2010 [1]. It conquered the world of web immediately, and has since been widely used in design of web pages and web-based applications.

In fact, this approach was, at least in parts and in some examples of web design, known earlier, but was not referred to as RWD. However, as Hans Selye wrote [2], the discovery “*is not to see something first, but to establish solid connections between the previously known and the hitherto unknown*”. From this point of view, Ethan Marcotte is generally recognized as the one who formulated the principles of the responsive web design and provided a general way to its implementation on the web.

The main idea of the RWD is the division of the content of a web page into blocks, which can be arranged in several logical ways in order to optimize the user experience. This, of course, reminds us the modular approach to typography [3]; however, instead of

*Corresponding author

a fixed-sized sheet of paper, a resizable browser window is the media for delivering the content. This different media prompts the modular approach to be enhanced by adding “fluidity” to it.

In this paper we take the idea of the “responsive design” further, apply it to the design of GUIs for stand-alone applications, and demonstrate how this can be done in MATLAB. The same technique can be used in other programming languages.

It should be mentioned that one can encounter some terms that sound similarly to the approach presented below in this paper, but they have different meaning. For instance, the term “responsive user interface” can in some contexts denote situations when the main process of an application is busy or not responding (frozen), while the GUI is still able to accept (to respond to) user’s inputs in the form of mouse clicks or keystrokes. Another such example is the term “adaptive user interface”. This term often describes situations when the user interface adapts to the role of a user, such as switching between basic and advanced modes, or switching between user roles such as “teacher/student”, “doctor/patient”, “admin/user”, etc.

2 Responsive GUI for applications

Recall that the responsive web design is based on the idea that a web page must respond by itself to the user’s behavior and the constraints of the media environment imposed by its screen size, pixel density, platform and orientation.

Hereby we extend this approach to designing the graphical user interface (GUI) of stand-alone applications, and arrive at what we call the *responsive graphical user interface* – ReGUI¹.

Before continuing, it is worth mentioning, that some elements of the responsive approach can already be observed in users’ interaction with computer programs.

First, many contemporary applications have the form of web applications. Their interface is a web page, so it is natural that for creating the user interface the RWD approach is used. Our focus, however, is on stand-alone applications.

Second, calculators for smartphones provide another example of responsive design – in portrait mode, a user sees a simple primary-school-like calculator, but rotating the smartphone to landscape transforms it into a scientific calculator with a different set of buttons and functions. Some other programs, like mobile web browsers, also provide similar functionality.

Third, touch keyboards in smartphones and tablets are also “responsive”, since switching to different languages changes the layout and the number of “touch keys”, additional keys appear specifically for particular applications, and so forth. This is somewhat similar to the behavior of aforementioned calculators for smartphones, because one whole set or subset of keys is shown while another is hidden.

¹Finding a suitable – and available – abbreviation is always a problem. . . We ended up with *ReGUI*; we found that in Catalan the word *regui* is one of the forms of the verb *regar* (‘to water’), which is well related to ‘fluidity’ of the responsive graphical user interface.

Also, in many desktop applications, like current word processors, spreadsheets, etc., separate document windows are opened or created in positions based on the number and physical dimensions of the computer screens available.

We define the basic requirements to the ReGUI in applications as follows:

- (R1) independence of the physical dimensions and resolution of the user’s device screen;
- (R2) ability to zoom in and out, preserving all proportions in the ReGUI elements, including fonts;
- (R3) responsiveness of the layout of the logical blocks and the elements of the ReGUI based on the user’s needs that are indirectly indicated by his/her behavior (resizing the application window, moving it, etc.).

Since we use MATLAB as a tool for scientific computations and applications in modelling and control of processes, we elaborated several examples of creating applications with different levels of ReGUI. Our choice of MATLAB is also based on the high readability of the MATLAB code.

The re-engineering of the “Matrix Poker” game [4], developed by the last author in 2003 in a fixed-size form with physical units of pixels, was done in October 2016 by consistently transforming all dimensions and font sizes to relative units (in MATLAB terminology, *normalized* units). The width-to-height aspect ratio is preserved, except for the situation, where the game window after resizing occupies the whole screen.

Creation of a simple zoomable application from scratch is easier, if we think in terms of relative (normalized) units from the very beginning. This is illustrated by the application Anaglyph3D [5] for making fine adjustments (micro-shifts and micro-rotations) of the left and right images of stereo photos that were taken separately using a single camera without a special dedicated tripod.

Both of these applications, Matrix Poker and Anaglyph3D, satisfy the first two requirements to the application’s ReGUI, i.e. requirements (R1) and (R2), but they do not contain enough logical blocks of the GUI elements to illustrate the core idea of ReGUI represented by the requirement (R3). How can one satisfy the requirement (R3) is shown in detail in the following example.

3 Creating a ReGUI application in MATLAB

The application TeachLCGE [6], that we describe below, has been developed for supporting teaching maxima and minima of functions of two variables. The source code is available at MATLAB Central File Exchange [6], and the video demonstration of its ReGUI functionality is available on YouTube [7].

The logical blocks that we consider in the user interface are:

- FUNCTION, with an editable text field for entering the formula for $f(x, y)$ and three buttons named **Show Example**, **Calculate**, **Clear the form**;

- STATIONARY POINTS;
- FUNCTION VALUES;
- LOCAL EXTREMA;
- PLOTS, with four subplots; the subplots can be arranged as 2x2 or 1x4;
- SHOW PLOTS IN SEPARATE WINDOWS, with four buttons – each button for a particular plot;
- a dedicated button BACK TO MAIN MENU, in which the user selects the type of the problem and the language (currently, English or Slovak).

These logical blocks are laid out in a different manner based on the width-to-height aspect ratio R of the resizable window of our application. In this example, we set up two breakpoints for changing the ReGUI layout, namely 0.75 and 1.5. When the aspect ratio R is between 0.75 and 1.5 (inclusive), the application window looks more or less similar to most computer screens with the old classical aspect ratio 4:3; when $R > 1.5$, then it is similar to wide screens in landscape orientation; and when $R < 0.75$, then it looks like a wide screen rotated 90 degrees to the portrait orientation.

It is important that our application window is fully resizable, so a user can change the width and the height of the application window independently, and as the result the layout of the all logical blocks and elements of the application is adapted to a new aspect ratio of the application window.

Let us very briefly outline the implementation of ReGUI. The first line numbers in the code fragments below correspond to the appropriate lines in our MATLAB application TeachLCGE [6].

Similar to the case of responsive web design, we start with defining the layout breakpoints (and, of course, mocking up the corresponding variants of the layout):

```
1043 % Layout aspect ratio breakpoints
1044     breakpoint = [0.75 1.5];
```

The application window aspect ratio is computed as usual:

```
1043 % Aspect Ratio
1044     appWinPosition = get(appWindow, 'Position');
1045     appWinWidth    = appWinPosition(3);
1046     appWinHeight   = appWinPosition(4);
1047     aspectRatio    = appWinWidth/appWinHeight;
```

In the next step, the data for the updated ReGUI layout is prepared (computed) based on the currently updated value of the aspect ratio:

```
1046 % ReGUI section
1047 if (aspectRatio >= breakpoint(1)) && (aspectRatio <=
    breakpoint(2))
```

```

1048     % Classic interface values
1049     panel0 = [0.01 0.75 0.38 0.175];
1050     titlePosition = [0.25 0.94 0.5 0.04];
1051     grsurfPosition = [0.01 0.935 0.12 0.04];
1052     % < ————— and so forth ————— >
1053 end
1054
1055 if (aspectRatio < breakpoint(1))
1056     % Tall interface values
1057     panel0 = [0.01 0.765 0.98 0.16];
1058     titlePosition = [0.05 0.96 0.9 0.04];
1059     grsurfPosition = [0.01 0.925 0.25 0.04];
1060     whitePosition = [0.25 0.93 0.33 0.04];
1061     % < ————— and so forth ————— >
1062 end
1063
1064 if (aspectRatio > breakpoint(2))
1065     % Wide interface values
1066     panel0 = [0.01 0.65 0.38 0.27];
1067     titlePosition = [0.05 0.93 0.9 0.06];
1068     grsurfPosition = [0.01 0.92 0.25 0.06];
1069     whitePosition = [0.10 0.935 0.12 0.05];
1070     % < ————— and so forth ————— >
1071 end

```

And finally the positions and/or the appearance of the ReGUI interface elements are updated using the computed data:

```

1095 if (Updater1 == 1)
1096     % Update values of properties of ReGUI elements
1097     set(frame0, 'Position', titlePosition);
1098     set(frame1, 'Position', examplePosition);
1099     set(frame2, 'Position', clearPosition);
1100     % < ————— and so forth ————— >
1101 end

```

The main steps that are described above are done by the function `regui`, which is triggered by the `ResizeFcn` property of the main application window:

```

247 % Main application window
248 appWindow = figure('Units', 'norm', ...
249     'ResizeFcn', @regui, ...
250     % < ————— and so forth ————— >
251     );

```

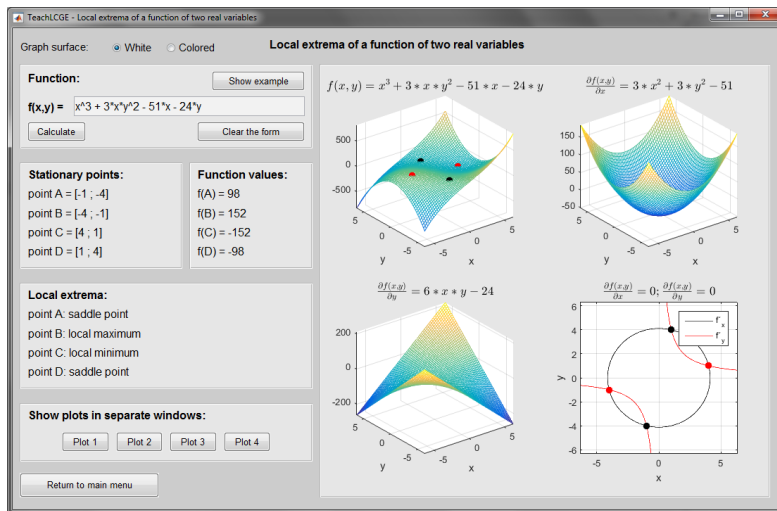


Figure 1: Layout for the aspect ratio between 0.75 and 1.5.

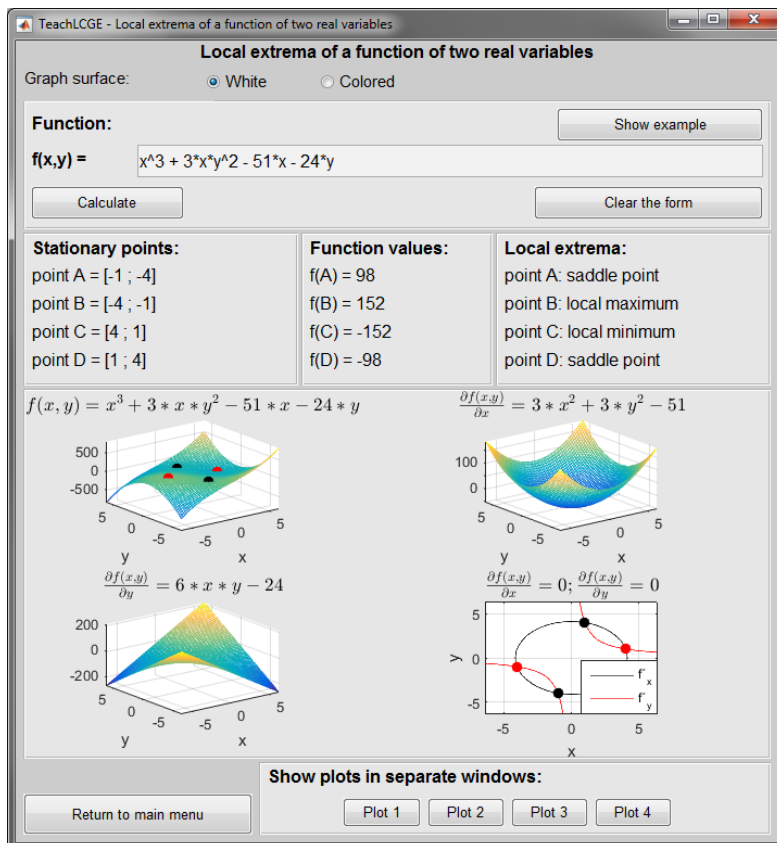


Figure 2: Layout for the aspect ratio less than 0.75.

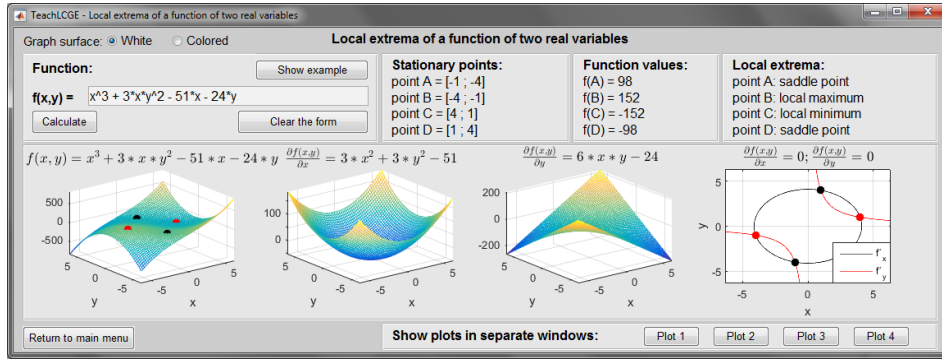


Figure 3: Layout for the aspect ratio greater than 1.5.

4 Conclusion

In this paper we extended the idea of responsive design for the case of stand-alone applications, and illustrated this on a sample application created in MATLAB. The same technique can be used in other programming languages as well.

Although in our example we – for the sake of simplicity – were changing only the layout of the logical blocks of the responsive graphical interface (ReGUI) of our sample application, it is also possible to hide or show selected blocks of the ReGUI based on the current size of the application window or certain proportions of its width and height. Further, it is possible to change the color depth of the elements of the ReGUI, their visibility, their background (for example, switching between a solid color and a background image), and other visual and functional properties of the elements of ReGUI. Also, in a similar manner the layout of the ReGUI of an application can change based on the position of the application on the computer screen – for example, swap the left- and right-sided logical blocks when a user moves the application window to the left or right edge of the screen.

Acknowledgments

This work was supported in parts by grants VEGA 1/0908/15, APVV-14-0892, SK-PL-2015-0038, ARO W911NF-15-1-0228.

References

- [1] Marcotte, E. Responsive Web Design. *A List Apart*, May 25, 2010. Available at <https://alistapart.com/article/responsive-web-design>. Accessed April 5, 2017.

- [2] Selye, H. *The Stress of Life*. Second edition. McGraw-Hill Education, New York, 1978.
- [3] West, S. *Working with Style: Traditional and Modern Approaches to Layout and Typography*. Watson-Guptill Publications, 1990.
- [4] Podlubny, I. *Matrix Poker*. Matlab Central File Exchange, submission ID 3168. March 21, 2003 (I. Podlubny); updated October 28, 2016 (M. Mikulsky). Available at <https://www.mathworks.com/matlabcentral/fileexchange/3168>. Accessed April 5, 2017.
- [5] Mikulsky, M. *Anaglyph3D*. Matlab Central File Exchange, submission ID 59537. October 6, 2016. Available at <https://www.mathworks.com/matlabcentral/fileexchange/59537>. Accessed March 17, 2017.
- [6] Mikulsky, M. *TeachLCGE*. Matlab Central File Exchange, submission ID 62480. 2017. Available at <https://www.mathworks.com/matlabcentral/fileexchange/62480>. Accessed April 9, 2017.
- [7] Mikulsky, M. *TeachLCGE – MATLAB Application & Responsive GUI Demonstration*. Available at <https://www.youtube.com/watch?v=ndYC11vIT2Y>. Accessed April 5, 2017.